Using VisTrails and Provenance for Teaching Scientific Visualization

Cláudio T. Silva, Erik Anderson, Emanuele Santos, and Juliana Freire

Scientific Computing and Imaging Institute
University of Utah

Abstract

Over the last 20 years, visualization courses have been developed and offered at universities around the world. Many of these courses use established visualization libraries and tools (e.g., VTK, ParaView, AVS, VisIt) as a way to provide students a hands-on experience, allowing them to prototype and explore different visualization techniques. In this paper, we describe our experiences using VisTrails as a platform to teach scientific visualization. VisTrails is an open-source system that was designed to support exploratory computational tasks such as visualization and data analysis. Unlike previous scientific workflow and visualization systems, VisTrails provides a comprehensive provenance management infrastructure. We discuss how different features of the system, and in particular, the provenance information have changed the dynamics of the Scientific Visualization course we offer at the University of Utah. We also describe our initial attempts at using the provenance information to better assess our teaching techniques and student performance.

1. Introduction

As the volume of digital data explodes, the ability to visualize these data has become increasingly important. More and more, scientists rely on visualizations to make sense out of data, from MRI and CT scans, water salinity measurements, to the results of computational experiments and simulations. Visualization is thus a topic that is essential in the education of the next generation of scientists and computer scientists.

Although the term "visualization" is somewhat recent, generally accepted to having been coined for the 1987 NSF report on scientific visualization [McC87], visualization has been used as a means of "data understanding by visual representation or other visual means" for hundreds of years. What separates the old from the new is the availability of advanced computing capabilities, including modern computer graphics techniques, which form the backbone of modern visualization research. While computer graphics techniques are an integral and important part of visualization, it is important to contrast the two fields. In the more traditional computer graphics, used to make movies or games, the goal is to produce visually engaging (beautiful) imagery that "appears plausible". In visualization, *accuracy* is more important than aesthetics. As such, users must be aware of errors and uncertainty present in the visualizations so that they are not misled by the resulting images [JMM*06]. Furthermore, the process of extracting insight from data goes beyond the generation of imagery, and needs to encompass the complete scientific discovery pipeline [van05, Shn07, Shn02]. Data exploration through visualization requires scientists to go through several steps. To successfully analyze and validate various hypotheses, it is necessary to pose several queries, correlate disparate data, and create insightful visualizations of both the simulated processes and observed phenomena.

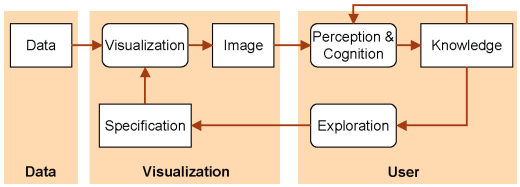

Figure 1: *A model for exploratory visualization. Adapted from J. van Wijk [van05].*

As illustrated in Figure 1, scientists need to assemble and execute complex visualization pipelines (workflows) that consist of data set selection, specification of series of operations that need to be applied to the data, and the creation of appropriate visual representations, before they can finally view and analyze the results. Often, insight comes from comparing the results of multiple visualizations created during the exploration process. For example, by applying a given visualization process to multiple datasets generated in different simulations; by varying the values of certain visualization parameters; or by applying different variations of a given process (e.g., which use different visualization algorithms) to a dataset.

The challenge of this exploration process is that for a visualization to be insightful, it needs to be both effective and efficient. This requires a combination of art, technology, and science to reveal information that is otherwise obscured. Despite the enormous progress in scientific visualization research, existing tools fail to support the analytical reasoning process that scientists use in their work. There is little or no support for linking data, visualizations, and knowledge. As insight is generated over time, the findings are not linked to supporting evidence, and scientific publications to a large extent stand on their own with little hard evidence of the scientific facts. To ensure result reproducibility, scientists often need to expend substantial effort managing data and their provenance. Provenance (also referred to as history, audit trail, lineage, or pedigree) captures information about the steps used to generate a given data product, be it a result or the computational task that produced it [SPG05, BF05, DF08, FKSS08, DBE*07]. Such information provides documentation that is key to preserving the data and determining the data's quality and authorship as well as interpreting, reproducing, sharing and publishing results. All of these are important requirements in the scientific process, and in some cases the provenance is as important as the results.

VisTrails and Data Exploration. VisTrails (`http://www.vistrails.org`) is an open-source system that was designed to support exploratory computational tasks such as visualization and data mining. A beta version of the VisTrails system was first released in January 2007. Since then, the system has been downloaded over twenty thousand times. VisTrails can be combined with existing tools and libraries, and provides comprehensive provenance management infrastructure. The availability of provenance information enables a series of operations which simplify exploratory processes and foster reflective reasoning [Nor94], for example: scientists can easily navigate through the space of workflows created for a given exploration task; visually compare workflows and their results; and explore large parameter spaces. The system also includes a series of usable interfaces for exploring the provenance information and supporting knowledge re-use.

Using VisTrails and Provenance in Teaching. Given our positive experience in deploying VisTrails to scientists, we decided to use the system as a platform for teaching scientific visualization. Our intuition was that the same features that support scientists in exploratory tasks would also be beneficial to students as they learn about visualization techniques. During Fall 2007 and Fall 2008, Professor Cláudio Silva used VisTrails for teaching the Visualization course at Utah. As we describe below, our experience with the course has shown that besides simplifying the creation of visualizations, the availability of provenance helps in other important aspects of the course. For instance, in the process of building examples of visualizations for the class, VisTrails allows the instructor to show the students not only the "final" result, but also the "path" he followed to derive the visualizations (the history tree)—including common mistakes that one makes in the process. During class, while responding to students' questions, it is possible to try out alternatives, and to show "differences" between them using both the visualization spreadsheet and the visual difference interface (Section 3). This makes the class more interactive and promotes active learning [act]. After the class, all the results and their provenance can be given to the students in the form of a vistrail, encoding the complete trail followed by the instructor while presenting examples and answering questions. The class notes are also accompanied by detailed provenance, allowing students to reproduce all examples. Another benefit of using VisTrails comes from the assignment provenance: instead of submitting *just* the final visualizations, students submit the complete history of the process they followed to create those visualizations. As we discuss in Section 4, this information can be very useful for the instructors, from helping them better assess their teaching effectiveness to identifying students in need of help.

2. Related Work

A quick search on the Web leads to a multitude of visualization courses being taught around the world. Many of the courses target not only computer science students, but also computational scientists and domain scientists from different disciplines. This is the case for the course at Utah. This choice alone has deep implications for the tools that are used for teaching. Although some courses use OpenGL as the basis for all the work, most courses make use of higher-level libraries, languages, and tools. The courses that rely mainly on OpenGL are often targeted to computer science students. Below, we describe some of the other tools used in courses.

The Application Visualization System (AVS) [UTFK*89] was one of the earliest and most influential visualization environments developed in the 1980s. It was based on a dataflow model and it was aimed at providing an easy to use, and powerful system for supporting the filter/map/render pipeline. The IBM Data Explorer (DX) [IBM] and the IRIS Explorer are two other systems from the same period. Testi-

mony to their effectiveness, these tools are still widely used today, over 20 years since they were originally developed. These dataflow-based visualizations systems can be seen as the precursors of current scientific workflow systems. As far as we could see, not many courses use these tools, and instead tend to use newer tools.

The early 1990s brought us Kitware's Visualization Toolkit (VTK) [SML03], which is an open source, object-oriented toolkit based on the dataflow programming model. For efficiency, the core components of the system are written in a compiled language (C++). For flexibility and extensibility, an interpreted language can be used for higher-level applications (Python, Tcl, or Java). This scripting capability and a large core set of algorithms has promoted VTK to its current status as one of the most popular visualization packages for researchers. VTK is widely used around the world, and many tools have been developed on top of it. Its open-source license has enabled the development of a number of influential end-user visualization tools. Many courses are based on VTK. Often, a high-level scripting language like Tcl or Python is used in those courses, since it is quite easy to teach the basics of these languages. Some courses that use VTK rely on higher-level tools for building the visualization pipelines. Two such tools developed are ParaView and VisIt. The ParaView project [LHA01] is aimed at not only extending VTK into a parallel framework, but also at developing a turnkey end-user tool that does not require users to explicitly build dataflow graphs. VisIt [CBB*05] has a similar goal, but in addition to VTK, it also integrates other data analysis libraries. Both systems can be scripted in Python.

Some other courses rely on integrated scientific computing environments, e.g. SCIRun [PJ95] and GRASPARC (GRAphical Support for PARallel Computing) [BPW*93]. These systems allow better integration of the overall computational pipeline instead of focusing only on visualizations.

3. VisTrails as a Teaching Tool

VisTrails is a freely-available system developed at the University of Utah that was designed to support exploratory computational tasks. A new concept introduced with VisTrails is the notion of provenance of workflow evolution [FSC*06]. In contrast to previous workflow and visualization systems which maintain provenance only for derived data products, VisTrails treats the workflows (or pipelines) as first-class data items and keeps their provenance.

As part of its provenance infrastructure, VisTrails provides usable interfaces for exploring the provenance information and supporting knowledge re-use [SFC07]. Students can take advantage of the detailed provenance accrued in examples to equip themselves and more easily tackle the visualization tasks required of them during the course. Since VisTrails provides utilities including query-by-example and refinement-by-analogy [SKV*07], students are able to quickly find and apply previously explored visualization pipelines to the task at hand.

VisTrails is an extensible system. Like other workflow systems, it allows pipelines to be created that combine multiple libraries. In addition, the VisTrails PythonSource construct[†] provides students with the ability to write arbitrary Python code to manipulate the input data. We leverage the extensibility of VisTrails to provide students with a collection of libraries particularly suited for scientific visualization. Established libraries such as VTK [SML06] and Matplotlib (`matplotlib.sourceforge.net`) provide some fundamental visualization components while libraries such as NumPy and SciPy [Oli07] allow students to more easily manipulate datasets as necessary.

Before we describe how we used VisTrails in the classroom, below we give a brief overview of the visualization course at the University of Utah.

3.1. The Visualization Course at Utah

Visualization courses throughout the world lack a comprehensive, widely-accepted text [RWE04]. At Utah, the first visualization course was offered over a decade ago. The current version of the course covers many standard topics, including the visualization pipeline, modeling data for visualization, elementary plotting techniques, color and human perception, 2-D visualization techniques, isosurfacing and volume rendering, information visualization, and aesthetics issues. Most of these topics take a week (two lectures), with isosurfacing and volume rendering taking the most time (three weeks total). We also had special lectures on introductions to geometry processing and computational topology as these topics are the basis for many newer visualization techniques. In previous years, "raw" VTK was used, and students would program in Tcl. Starting in 2007, students used VisTrails to build their visualizations [‡].

We use a number of different datasets to give students exposure to multiple data types. We use both acquired and simulated data, and include data that requires different modeling primitives (scattered points, structured and unstructured data representing scalar, vector, and tensor fields). Each student in the visualization course is required to complete six separate, and increasingly complex, tasks using VisTrails, VTK, and matplotlib. The last assignment is open ended. Students were asked to create visualizations of the cosmology data displayed in Figures 2 and 4 (from Los Alamos National Laboratory [AAH*08]) in the last homework assignment in Fall 2007. Although students are not required to implement

[†] http://sourceforge.net/projects/vistrails/files/vistrails/vistrails-usersguide-1.3-rev198.pdf

[‡] Full course material for the 2007 and 2008 classes are available at http://www.vistrails.org/index.php/SciVisFall2007 and http://www.vistrails.org/index.php/SciVisFall2008, respectively.

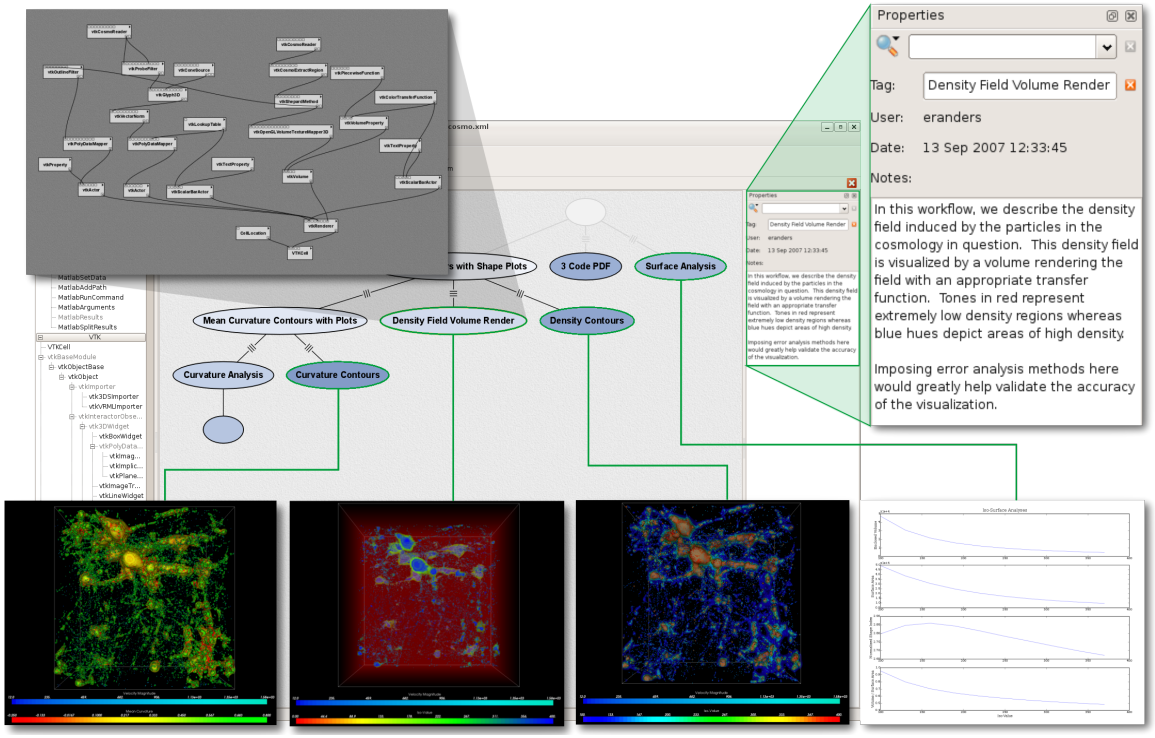

Figure 2: *An example of an exploratory visualization for studying celestial structures derived from cosmological simulations using VisTrails [AAH*08]. Complete provenance of the exploration process is displayed as a vistrail (history tree) with each node representing a workflow that generates a unique visualization. Detailed meta-data is also stored including free-text notes made by the student, the date and time the workflow was created or modified, optional descriptive tags, and the user that created it.*

each visualization algorithm (since they can use VTK), the understanding of each technique is required to arrive at adequate visualizations.

As we discuss below, our initial assessment indicates that VisTrails has allowed the students to focus on the visualization tasks, instead of having to spend substantial effort developing user interfaces. Besides simplifying the construction of pipelines, the provenance mechanisms also streamline the exploratory process required to produce the visualizations, and enhance interactions between students, instructor, and teaching assistants.

3.2. Using Provenance in the Classroom

A key feature of VisTrails that distinguishes it from other visualization and scientific workflow systems is its ability to capture the evolution history of a workflow's specification. VisTrails accomplishes this through a change-based provenance mechanism that uniformly captures changes to parameter values as well as workflow definitions [FSC*06, CFS*06]. By maintaining detailed provenance of the exploration process in this way, VisTrails ensures reproducibility both by the student and the grading teaching assistant. Additional benefits are realized by instructors inspecting the work habits of the students both during the semester as well as after the completion of the course (Section 4).

Providing students with the provenance gathered during the examples covered in class allows them to reproduce the examples as well as to experiment with variations, contributing to a better understanding of fundamental properties of various visualization techniques. Students have traditionally responded positively to this method of instruction as it allows them to explore the advantages and disadvantages of different techniques more easily.

3.3. Interacting with Provenance

Figure 2 shows an example of several visualizations created using VisTrails. In the center, the history tree (or vistrail) captures all modifications students apply to their visualizations. Each node in this tree corresponds to a workflow (or pipeline) while edges between the nodes represent changes applied to transform the parent pipeline into the child (e.g., through the addition of a module or a change to a parame-

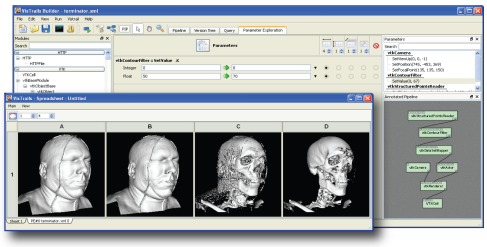

Figure 3: *Parameter exploration is performed in VisTrails using a simple interface. The results are computed efficiently by avoiding redundant computation and displayed in the spreadsheet for interactive comparative visualization.*

ter value). The tree-based representation allows students to return to previous versions in an intuitive way. This interaction enables mechanisms to undo incorrect changes, to form comparisons between different workflows, as well as to create visual difference that highlights the actions leading to a particular result. Furthermore, by later viewing the series of steps students take to realize a visualization, instructors are able to more easily gauge the efficacy of the class lectures. This allows more efficient teaching strategies to be employed as the needs of the classroom change over the semester.

3.4. Comparing Pipelines and Visualizations

VisTrails' change-based provenance model also enables operations that simplify the derivation and comparison of multiple data products [FSC*06]. Since the discovery process requires many trial-and-error steps, it is common for tens to hundreds of different workflows and parameterizations to be explored in the course of creating a single visualization. This exploration of a parameter space both during and after the visualization is created is important to the development of insights about the data being studied.

VisTrails' spreadsheet-based visualization mechanism enables the direct comparison of multiple visualizations. Figure 3 illustrates the use of the spreadsheet for the comparison of different parameter values for isosurface extraction. Comparisons made in this way allow students to more quickly arrive at appropriate parameterizations from everything from iso-value choices to transfer function design.

In addition to direct comparison of visualizations, VisTrails also enables the comparison of the *workflows that generated the visualizations*. This visual workflow difference, as illustrated in Figure 4, allows a student to rapidly determine the most appropriate visualization method to highlight salient aspects of the data in question. In this figure, the visual difference shows modules unique to the different workflows by color, while rendering modules shared by the workflows in gray. By varying the lightness of shared modules, users can easily detect modules with different parameterizations.

Task	Description	Difficulty	Open-Endedness
Task 1	Introduction	1	1
Task 2	2D Visualization Techniques	3	2
Task 3	Scalar & Vector Field Visualization	3	2
Task 4	Isosurfacing & Volume Rendering	4	3
Task 5	Diffusion Tensor Imaging & InfoVis	4	4
Task 6	Open-Ended Visualization	5	5

Table 1: *Description of the six tasks involved in the study with the instructor's expectation of difficulty and open-endedness on a scale from 1 to 5.*

4. Learning from Provenance

An added benefit of using VisTrails for teaching is that instructors can collect and analyze the provenance of students' work. Instead of submitting only the final visualizations, students hand-in the provenance of their assignments: all the trial-and-error steps they followed to complete the assignment. This information can help the instructor gain insight into various aspects of the course. For example, the provenance makes it possible to analyze, in an unobtrusive manner, different approaches to workflow design as well as common usage patterns [HMSA08]. It can also aid an instructor better assess their teaching effectiveness and identify students who need help. In this section, we present a preliminary analysis of the provenance generated by students taking the Visualization course in the Fall of 2007 (http://www.vistrails.org/index.php/SciVisFall2007).

4.1. The Data

A total of thirty students took the course. Throughout the semester, they were assigned six different tasks with fixed deadlines. Table 1 provides a short description as well as a subjective evaluation by the course instructor of the difficulty and open-endedness of each task.

Students used VisTrails to complete the tasks and for each task, they submitted a file containing all the actions they performed. These actions were transparently captured by VisTrails and stored according to the change-based model. Each action has a unique identifier; the identifier of its parent action; the user who performed the action; a timestamp indicating when the action took place; an optional tag; free-text

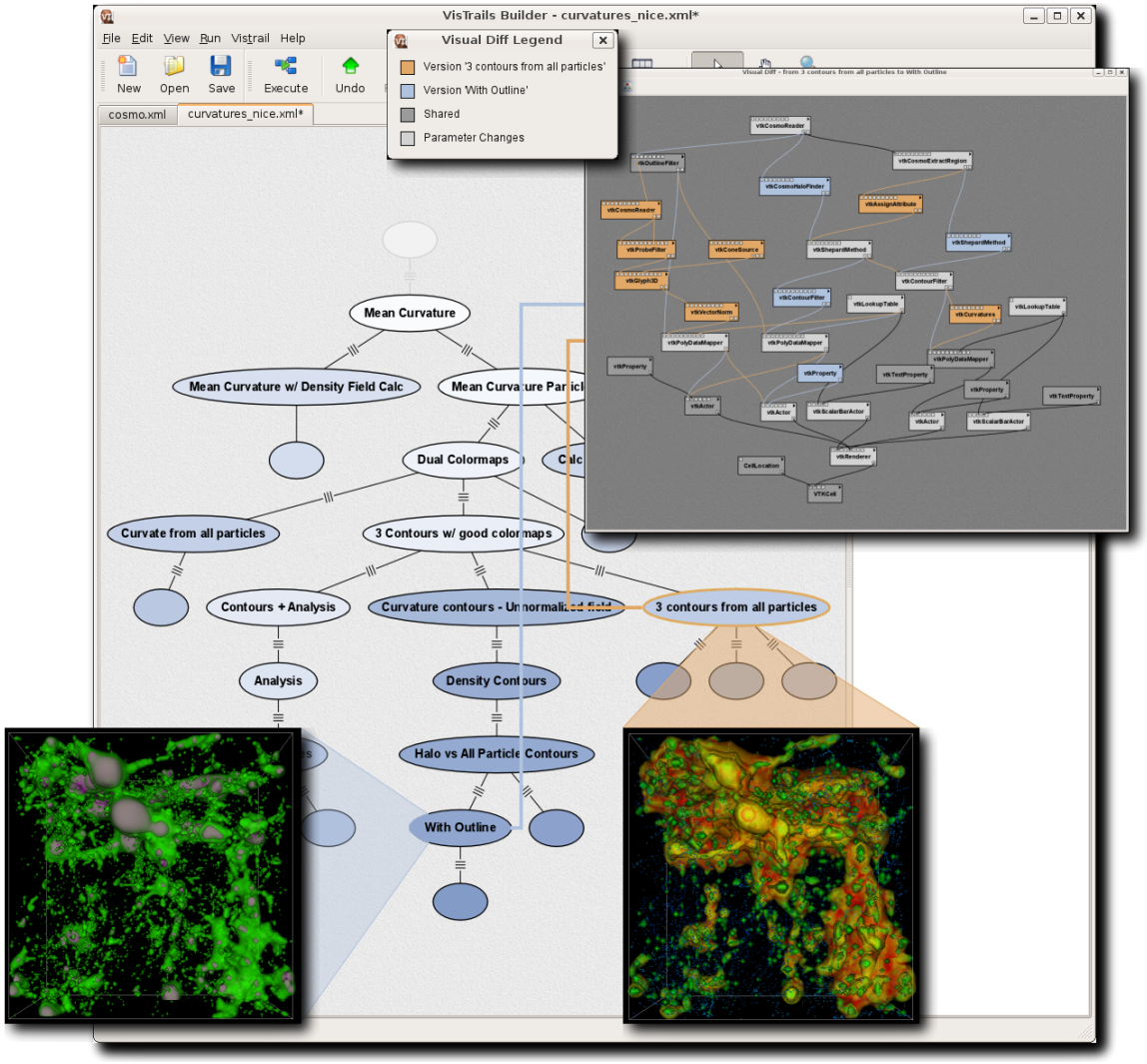

Figure 4: *By representing provenance as a series of actions that modify a pipeline, visualizing the differences between two workflows becomes possible. The difference between workflows is described in a meaningful way, as an aggregation of the two workflows forming it. This representation of the difference is both informative and intuitive, reducing the time it takes to understand how two workflows are functionally different.*

annotations; and the required information to reproduce the action.

4.2. Analyzing Evolution Provenance at Different Levels

Because our provenance data encompasses a range of tasks completed by a set of users, it can be analyzed in different levels. Globally, we can observe trends across all tasks and users. At the task level, we can attempt to characterize tasks by the types of actions involved. Finally, for a specific user, we can drill down to assess progress, working habits, and strategies used for different tasks.

Because we know exactly when each action occurred, it is possible to plot the total workload against time. The activity histogram in Figure 5 shows that, unsurprisingly, most work was condensed into the few days preceding the task deadlines. Besides that, the activity histogram also gives a good sense of which tasks required more effort. Although this measure may not match the assessment of the instructor, it gives a better measure of the effort the students employed.

4.2.1. Global Analysis

One useful feature of workflow evolution provenance is that users can interact with the provenance as they work. For ex-

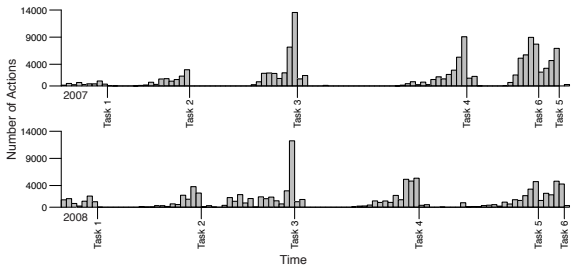

Figure 5: *Activity histogram of action dates with due dates indicated for both 2007 and 2008 classes.*

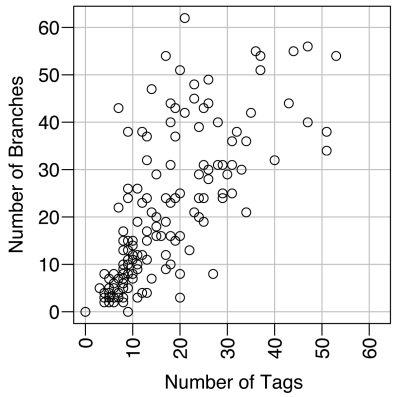

Figure 6: *The correlation between the number of branches and the number of tags per user-task.*

ample, users can at any time access the version tree and select any existing workflow to execute it, to inspect its specification or to modify it. In this last case, a new *branch* with the modified workflow specification is created as a new leaf of the tree. In order to help users to identify workflow specifications, VisTrails allows them to *tag* the nodes in the tree. In our analysis, we found that the number of branches in the version tree is correlated with the number of tagged nodes, as shown in Figure 6. This indicates that, as users have to revisit a previously defined workflow, they would select a tagged node because it is easier to identify.

4.2.2. Analysis of Tasks

Workflow evolution information can also be helpful to characterize tasks. As noted in Table 1, the tasks assigned to the scientific visualization students varied in their goals, difficulty, due date, and how open-ended they were. To illustrate how workflow evolution data can be used to understand the different types of work involved in a task, we classified the actions involved in workflow development into: *structural actions* (addition and deletion of modules and connections in the workflow); *parameter actions* (modification of parameter values in the workflow); and *layout actions* (changes to the locations of modules in visual programming interface).

Figure 7 shows an attempt to characterize tasks by the types of actions involved. For all users, we calculated the overall percentage of actions that were structural, parameter and layout actions across all tasks (Figure 7(a)). In addition, we computed these percentages for each task, as shown in Figure 7(b), (c) and (d). The distributions of these percentages were plotted as boxplots. Note that the percentage of actions spent changing parameters has the greatest variance for most tasks. This should be expected as some users locate correct parameter values faster than others, and some will also expend more effort tweaking parameters than others. Another interesting feature of these plots is that Task 5 shows more structural activity than Tasks 2, 3, and 4. This is explained by the fact that students were given examples for the previous three tasks, and in Task 5, they were left to discover how to create workflows from scratch.

4.2.3. Analysis of Users

A useful application of workflow evolution provenance is to help in understanding how different users approach a problem. Figure 8 shows two trees created by different users for the same task. User 1 and User 2 clearly have different development styles: the tree derived by User 2 is both shorter and narrower than that of User 1. This figure also shows a plot of the branching factor of the version trees across the tasks for User 1 and User 2. A smaller branching factor indicates that a more direct path was used to obtain a solution. In contrast, a larger branching factor indicates that more trial-and-error steps were followed. There are many cases where branching can be useful, including when a user wishes to develop workflows that share a common subworkflow: the user designs the first workflow, goes to the version tree, selects the node corresponding to the common subworkflow and from there branches to the second workflow. We found a range of branching factors that varied across users and tasks.

Branching is just one variable from the workflow evolution provenance data that can be used to identify "user signatures", other variables, such as the time between actions and the number of sessions may also lead to insights in this respect.

5. Discussion

We strongly believe that teaching is one of the killer applications of provenance-enabled systems. Provenance information can help instructors to be more effective and improve the students' learning experience. Due to the provenance information, it is possible for one person to see what another person did, and to easily compare their own work to it. This makes it possible for the instructors to share their own work with the students, who can easily see how the problem was approached by someone with more experience. When making new functionality available (e.g., a new VTK module),

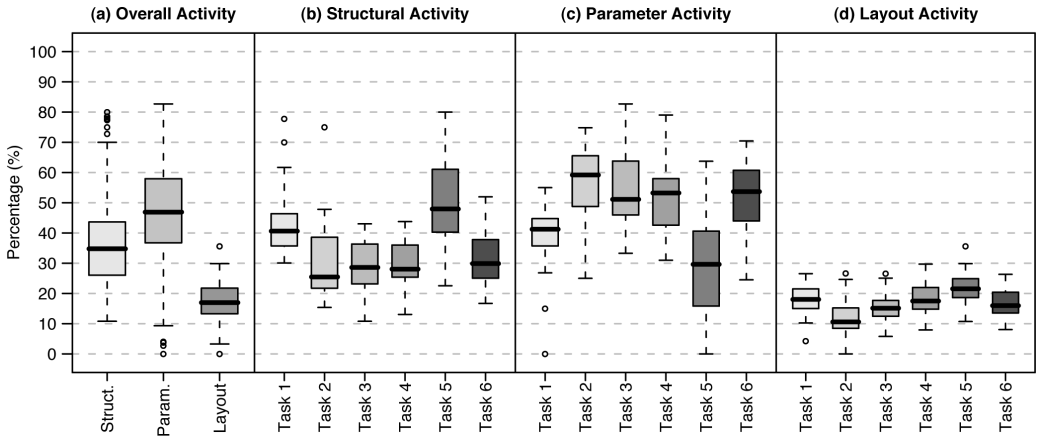

Figure 7: *Structural, Parameter and Layout Activity of Workflows.*

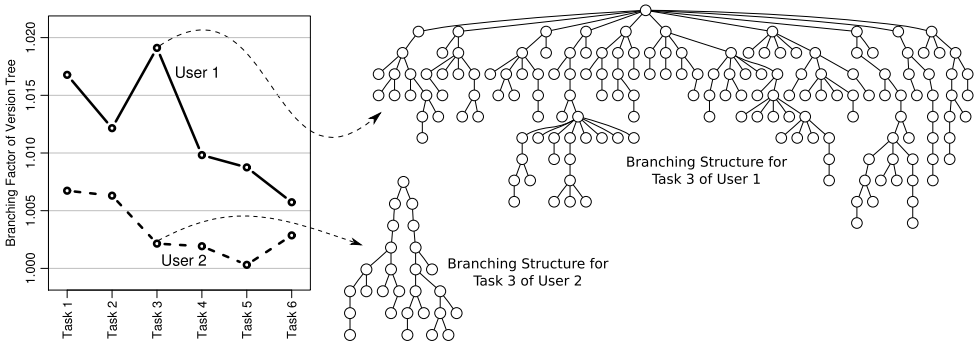

Figure 8: *Plot of Branching Factors for the six tasks from two different users. The branching structure for Task 3 is depicted on the right.*

the process of using the new module in an example can easily be turned into a tutorial on how to use the new functionality. This also makes it easier to have adoption in other places. An important benefit of the unobtrusive way that VisTrails captures provenance is that there is no extra burden on the user; the user can do her work as usual and the tool transparently records all of her actions, which she can revisit later. The tree-based representation for the provenance allows the user to return to a previous version in an intuitive way, to undo bad changes, to compare different pipelines, and to be reminded of the actions that led to a particular result.

The data in the previous section shows that workflow evolution provenance allows one to measure, summarize, and analyze new aspects of workflow specification and design. A detailed analysis of how time is spent in workflow design can help to provide an understanding of how users interact with workflow systems. In addition, these statistics can produce insights into the potential bottlenecks and how these systems can be improved. While our results represent only an initial examination, we have discovered a number of areas where comparative statistics offer a window into general workflow design patterns, task characterization, and exploratory styles.

In the course of our study, we have identified some limitations of the VisTrails provenance capture mechanism. We plan to improve and augment the variables captured by the change-based model to allow for more accurate and detailed analyses. Specifically, while each change is time-stamped, it is difficult to determine the actual time involved in performing a single action. In addition, information about distinct sessions of work would be useful to better determine the actual time spent accomplishing the computational tasks. There are also some actions that are not captured by the system, including navigation over the version tree (e.g., a user backtracking to different nodes), which can be useful to identify problem-solving patterns as well as students that might be confused. Last but not least, although VisTrails captures the provenance information, analyzing it can be challenging. We are currently developing a provenance analytics toolkit that provides a visual interface for basic an-

alytics operations and allows users to interactively explore the information.

In our initial analysis, we just examined the provenance derived by the students. We would also like to cross quality or merit data about the pipeline specifications with the provenance data to infer information about which practices led to good pipeline specification and how time was used in these cases. Also, we considered only general actions for modifying pipelines. In future work, we plan to perform analyses that take into account the semantics of the individual actions. For example, instead of looking at the addition and deletion of modules, for a visualization task, we could consider the addition of a volume renderer or of an isosurface extraction. By doing so, we could measure the effort involved in applying these two different visualization techniques.

6. Conclusions

In this paper we report on our experiences teaching the visualization course using a provenance-enabled visualization tool. Due to space limitations, we did not report on the course reviews, which were very positive. The comments from the students support our intuition that using provenance for teaching can have a positive effect on learning. Based on this experience, we would like to further explore this paradigm and take this type of provenance-enabled teaching to the next level. We believe we need to improve our "provenance analytics" tools to take full advantage of our approach. In particular, as discussed in Section 5, by analyzing the provenance of the students' work, instructors can learn useful information that can help them improve the course and better guide the students.

Acknowledgments

We thank Steve Callahan, Carlos Scheidegger and Lauro Lins for their help with the scientific visualization course in 2007 and 2008 at the University of Utah. We also thank the students that provided their vistrail files for this study. Our research has been funded by the National Science Foundation (grants IIS-0905385, IIS-0746500, ATM-0835821, IIS-0844546, CNS-0751152, IIS-0713637, OCE-0424602, IIS-0534628, CNS-0514485, IIS-0513692, CNS-0524096, CCF-0401498, OISE-0405402, CCF-0528201, CNS-0551724), the Department of Energy SciDAC (VACET and SDM centers), and IBM Faculty Awards (2005, 2006, 2007, and 2008). E. Santos is partially supported by a CAPES/Fulbright fellowship.

References

[AAH*08] ANDERSON E. W., AHRENS J., HEITMANN K., HABIB S., SILVA C.: Provenance in comparative analysis: A study in cosmology. *Computing in Science and Engineering 10*, 3 (2008), 30–37.

[act] Active Learning. http://en.wikipedia.org/wiki/Active_learning.

[BF05] BOSE R., FREW J.: Lineage retrieval for scientific data processing: a survey. *ACM Computing Surveys 37*, 1 (2005), 1–28.

[BPW*93] BRODLIE K., POON A., WRIGHT H., BRANKIN L., BANECKI G., GAY A.: Grasparc: a problem solving environment integrating computation and visualization. In *IEEE Visualization 1993* (1993), pp. 102–109.

[CBB*05] CHILDS H., BRUGGER E. S., BONNELL K. S., MEREDITH J. S., MILLER M., WHITLOCK B. J., MAX N.: A contract-based system for large data visualization. In *IEEE Visualization 2005* (2005), pp. 190–198.

[CFS*06] CALLAHAN S., FREIRE J., SANTOS E., SCHEIDEGGER C., SILVA C., VO H.: Managing the evolution of dataflows with vistrails *(Extended Abstract)*. In *IEEE Workshop on Workflow and Data Flow for Scientific Applications (SciFlow)* (2006).

[DBE*07] DAVIDSON S. B., BOULAKIA S. C., EYAL A., LUDÄSCHER B., MCPHILLIPS T. M., BOWERS S., ANAND M. K., FREIRE J.: Provenance in scientific workflow systems. *IEEE Data Eng. Bull. 30*, 4 (2007), 44–50.

[DF08] DAVIDSON S. B., FREIRE J.: Provenance and scientific workflows: challenges and opportunities. In *Proceedings of ACM SIGMOD* (2008), pp. 1345–1350.

[FKSS08] FREIRE J., KOOP D., SANTOS E., SILVA C. T.: Provenance for computational tasks: A survey. *Computing in Science and Engineering 10*, 3 (2008), 11–21.

[FSC*06] FREIRE J., SILVA C. T., CALLAHAN S. P., SANTOS E., SCHEIDEGGER C. E., VO H. T.: Managing rapidly-evolving scientific workflows. In *International Provenance and Annotation Workshop (IPAW)* (2006), LNCS 4145, pp. 10–18.

[HMSA08] HEER J., MACKINLAY J., STOLTE C., AGRAWALA M.: Graphical histories for visualization: Supporting analysis, communication, and evaluation. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (Nov–Dec 2008), 1189–1196.

[IBM] IBM: OpenDX. http://www.research.ibm.com/dx.

[JMM*06] JOHNSON C. R., MOORHEAD R., MUNZNER T., PFISTER H., RHEINGANS P., YOO T. S.: *NIH-NSF Visualization Research Challenges*. IEEE Computer Society, 2006. http://tab.computer.org/vgtc/vrc/index.html.

[LHA01] LAW C. C., HENDERSON A., AHRENS J.: An application architecture for large data visualization: a case study. In *IEEE Symposium on Parallel and Large-data Visualization and Graphics 2001* (2001), pp. 125–128.

[McC87] Visualization in scientific computing. In *Computer Graphics*, McCormick B., DeFanti T., Brown M., (Eds.), vol. 21. 1987.

[Nor94] NORMAN D. A.: *Things That Make Us Smart: Defending Human Attributes in the Age of the Machine*. Addison Wesley, 1994.

[Oli07] OLIPHANT T. E.: Python for scientific computing. *Computing in Science and Engineering 9*, 3 (2007), 10–20.

[PJ95] PARKER S. G., JOHNSON C. R.: SCIRun: a scientific programming environment for computational steering. In *Proceedings of Supercomputing* (1995), p. 52.

[RWE04] ROTARD M., WEISKOPF D., ERTL T.: Curriculum for a course on scientific visualization. *ACM SIGGRAPH Workshop on Computer Graphics Education* (2004).

[SFC07] SILVA C., FREIRE J., CALLAHAN S. P.: Provenance for visualizations: Reproducibility and beyond. *IEEE Computing in Science & Engineering 9*, 5 (2007), 82–89.

[Shn02] SHNEIDERMAN B.: Acm's computing professionals face new challenges. *Commun. ACM 45*, 2 (2002), 31–34.

[Shn07] SHNEIDERMAN B.: Creativity support tools: accelerating discovery and innovation. *Commun. ACM 50*, 12 (2007), 20–32.

[SKV*07] SCHEIDEGGER C., KOOP D., VO H., FREIRE J., SILVA C.: Querying and creating visualizations by analogy. *IEEE Transactions on Visualization and Computer Graphics* (2007).

[SML03] SCHROEDER W., MARTIN K., LORENSEN B.: *The Visualization Toolkit An Object-Oriented Approach To 3D Graphics*. Kitware, 2003.

[SML06] SCHROEDER W., MARTIN K., LORENSEN B.: *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Kitware, Inc., 4th Edition, 2006.

[SPG05] SIMMHAN Y. L., PLALE B., GANNON D.: A survey of data provenance in e-science. *SIGMOD Record 34*, 3 (2005), 31–36.

[UTFK*89] UPSON C., THOMAS FAULHABER J., KAMINS D., LAIDLAW D. H., SCHLEGEL D., VROOM J., GURWITZ R., VAN DAM A.: The application visualization system: A computational environment for scientific visualization. *IEEE Computer Graphics and Applications 9*, 4 (1989), 30–42.

[van05] VAN WIJK J.: The value of visualization. In *Proceedings of IEEE Visualization* (2005).